



**LAPPEENRANTA UNIVERSITY OF TECHNOLOGY**  
**Department of Electrical Engineering**

**MASTER'S THESIS**

**MUON DETECTOR LINK SYSTEM TEST SET-UP**

The topic of the Master's thesis has been approved by the department council of the Department of Electrical Engineering on 9 November 2005.

The supervisors and examiners of the thesis are Professor Tuure Tuuva and M.Sc. (Tech.) Matti Iskanius.

Lappeenranta 21.11.2005

Vesa Väisänen  
Korpimetsäntie 5 C 2  
53850 Lappeenranta  
Finland  
+358 50 413 2581

## TIIVISTELMÄ

Tekijä: Vesa Väisänen  
Työn nimi: Myoni-ilmaisimen linkkijärjestelmän testiympäristö  
Osasto: Sähkötekniikka  
Vuosi: 2005  
Paikka: Lappeenranta

Diplomityö. Lappeenrannan teknillinen yliopisto. 53 sivua, 31 kuvaa, 3 taulukkoa ja 4 liitettä.

Tarkastajat: Professori Tuure Tuuva ja DI Matti Iskanus

Hakusanat: myoni, CMS, RPC, DCS

CERNin tutkimuskeskuksen rakenteilla olevan hadronikiihdyttimen eräs tarkoitus on todistaa Higgsin bosonin olemassaolo. Higgsin bosonin löytyminen yhtenäistäisi nykyisen hiukkasfysiikan teorian ja antaisi selityksen sille kuinka hiukkaset saavat massansa. Kiihdyttimen CMS koeasema on tarkoitettu erityisesti myonien ilmaisuun.

Tämä työ liittyy CMS koeaseman RPC-ilmaisintyyppin linkkijärjestelmään, jonka tarkoituksena on käsitellä ilmaisimelta tulevia myonien aiheuttamia signaaleja ja lähettää tiedot tärkeäksi katsotuista törmäystapahtumista tallennettavaksi analysointia varten. Työssä on toteutettu linkkijärjestelmän ohjaus- ja linkkikorteille testiympäristö, jolla voidaan todeta järjestelmän eri osien keskinäinen yhteensopivuus ja toimivuus. Työn alkuosassa esitellään ilmaisimen linkkijärjestelmän eri osat ja niiden merkitykset. Työn loppuosassa käydään läpi eri testimenetelmiä ja analysoidaan niiden antamia tuloksia.

## **ABSTRACT**

Author: Vesa Väisänen  
Title: Muon detector link system test set-up  
Department: Electrical Engineering  
Year: 2005  
Place: Lappeenranta

Master's Thesis. Lappeenranta University of Technology. 53 pages, 31 figures, 3 tables and 4 appendices.

Supervisors: Professor Tuure Tuuva and M.Sc. (Tech.) Matti Iskanius

Keywords: muon, CMS, RPC, DCS

One purpose of the CERN Large Hadron Collider is to prove the existence of the Higgs particle. The proven existence of the Higgs particle would unite the present theory of particle physics and would explain how the particles get their masses. The CMS experiment station of the collider is dedicated for muon detection.

This work is related to RPC muon detector link system in the CMS station. The purpose of the RPC link system is to gather the muon-triggered signals coming from the detector chambers and to send the available information of important events for further analysis. A test set-up for the control and link boards of the link system has been developed. It can be used to test, that the different parts of the system are functional and compatible with each other. The purposes of the different system components are described in the first part of the work and the different test methods and their results are analyzed at the end.

## **FOREWORD**

This Master's thesis has been done as a part of the CERN projects of Microelectronics laboratory and Electronics Design Centre at Lappeenranta University of Technology.

I want to thank Professor Tuure Tuuva for giving me an opportunity to take part in this project and M.Sc. Matti Iskanius and other staff from Electronics Design Centre for many practical advices. I am also thankful for all my friends and colleagues for encouraging and good company.

Special thanks go to my parents and my brother and sister, who have helped and supported me through all these years. Thank God for the miracle of life.

Lappeenranta 21.11.2005

Vesa Väisänen

## TABLE OF CONTENTS

Symbols and abbreviations .....	3
1. Introduction .....	6
1.1 Background .....	6
1.2 The Compact Muon Solenoid (CMS) .....	7
1.2.1 The Pixel Detector.....	7
1.2.2 The Silicon Tracker.....	8
1.2.3 Electromagnetic and Hadronic Calorimeters .....	8
1.2.4 The Muon Detection System.....	8
1.3 Purpose of the work .....	9
2. Parts of the setup .....	10
2.1 Front-end electronics.....	10
2.2 The Link Board box .....	12
2.2.1 The Link Board .....	14
2.2.2 The Control Board.....	15
2.3 Timing, Trigger and Control system.....	16
2.3.1 The VMEbus .....	20
2.3.1 TTCvi .....	23
2.3.2 TTCvx .....	26
2.4 The Detector Control System.....	27
2.4.1 DCS overview .....	27
2.4.2 DCS hardware .....	28
2.4.3 DCS software .....	30
3. The test set-up .....	33
3.1 The TTC system and the Front-End Controller .....	33
3.1.1 Testing of the TTCvi and TTCvx.....	34
3.1.2 Testing of the front-end controller .....	39
3.2 Programming of the CBIC .....	40
3.3 Programming of the LBC and Syncoder.....	42
3.4 The CCU ring.....	46
3.5 Reading test pulse data from the link board.....	47
3.6 The link board tester.....	48

4. Conclusions .....	50
References .....	52

Appendix 1: Front panel of the TTCvi.

Appendix 2: Front panel of the TTCvx.

Appendix 3: Test results from CCU ring test.

Appendix 4: Test results from test pulse readout.

## **Symbols and abbreviations**

AC	Alternating Current
ADO	Address Only
ADOH	Address Only with Handshake
ALICE	A Large Ion Collider Experiment
ARM	Abbreviation for a processor core type (Acorn RISC Machine)
ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
ATLAS	A Toroidal LHC Apparatus
BLT	Block Transfer
CB	Control Board
CBIC	Control Board Initialization Controller
CBPC	Control Board Programmable Controller
CCU	Communication and Control Unit
CERN	European Organization for Nuclear Research
CMS	Compact Muon Solenoid
CR	Configuration Read-only Memory
CSC	Cathode Strip Chamber
CSR	Configuration and Status Registers
DAQ	Data Acquisition
DC	Direct Current
DCS	Detector Control System
DIP	Dual In-line Package
DOH	Digital Optohybrid
ECAL	Electromagnetic Calorimeter
ECL	Emitter Coupled Logic
EMC	Electromagnetic Compatibility
ESD	Electrostatic Discharge
FEB	Front-End Board
FEC	Front-End Controller
FPGA	Field Programmable Gate Array
GOL	Gigabit Optical Link

HAL	Hardware Access Library
HCAL	Hadronic Calorimeter
I <sup>2</sup> C	The Inter-IC bus
IMS	Information Message Service
JTAG	Joint Test Action Group
L1A	Level-1 Accept
LBC	Link Board Controller
LBUS	Local Bus
LBx	Link Board Box
LCK	VMEbus lock
LED	Light Emitting Diode
LEMO	Push-pull self-latching connector named after the founder of the LEMO company, engineer Léon Mouttet
LEP	Large Electron Positron Collider
LHC	Large Hadron Collider
LHCrx	Receiver module for the LHC clock
LV	Low Voltage
LVDS	Low Voltage Differential Signaling
MBLT	Multiplexed Block Transfer
MLB	Master Link Board
NIM	Nuclear Instrumentation Module
PAC	Pattern Comparator
PIN	Positive-Intrinsic-Negative
PIO	Parallel Input/Output
PLL	Phase Locked Loop
PROM	Programmable Read-Only Memory
RB	Readout Board
RMS	Root Mean Square
RMW	Read Modify Write
RPC	Resistive Plate Chamber
SCL	Serial Clock
SDA	Serial Data
SLB	Slave Link Board



SPS	Super Proton Synchrotron
TB	Trigger Board
TIB	Tracker Inner Barrel
TID	Tracker Inner Disks
TriDAS	Trigger and Data Acquisition
TTC	Timing, Trigger and Control System
TTCcf	TTC clock fanout module
TTCex	Laser encoder/transmitter module
TTCmi	TTC minicrate
TTCmx	Laser transmitter in the minicrate
TTCox	Optical TTC tree coupler
TTCrx	ASIC to receive and decode the TTC signal
TTCtx	Laser transmitter using VMEbus
TTCvi	TTC interface to VMEbus
TTCvx	Module for TTC encoding with four optical LED transmitters
VCSEL	Vertical Cavity Surface-Emitting Laser
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VME	VERSAmodule Eurocard. Standard architecture for bus and mechanics
XDAQ	Framework for data acquisition programs
XML	Extensible Markup Language

# 1. Introduction

## 1.1 Background

There has been enormous progress in the understanding of the basic composition of matter during the last decades. We know that matter is made of molecules, molecules from atoms, atoms from electrons, protons and neutrons and furthermore these are made from quarks etcetera. The big question at the moment is how the particles get their masses. Why are the masses what they are? Why are the ratios of masses what they are? Answers to these questions could be found from the Higgs mechanism. According to the Standard Model of particle physics, the vacuum in which all particle interactions take place is not actually empty, but is instead filled with a condensate of Higgs particles. There exist continuous collisions between quarks, leptons, and W and Z bosons and Higgs particles as they travel through the "vacuum". The Higgs condensate acts like molasses and anything that interacts with it is slowed down. The particles become heavier when they interact with the Higgs condensate. The stronger the interactions the heavier the particles become. [1]

The building of the Large Hadron Collider (LHC) has been greatly motivated by the Higgs particle. The LHC is being installed in 27-kilometer ring deep below the countryside on the outskirts of Geneva, Switzerland and should be operational by 2007. There are four main experiment sites where the collisions will be analyzed: ALICE, ATLAS, CMS and LHC. This work is related to the CMS RPC trigger system.

## 1.2 The Compact Muon Solenoid (CMS)

The abbreviation CMS comes from Compact Muon Solenoid. The word compactness derives from the structure of the system. The heart of the CMS is a very high field solenoid magnet, which is surrounded by a massive iron yoke. The overall diameter is 14.60 m and overall length is 21.60 m. The muons are detected by their bending in a very high magnetic field, which intensity can be up to 4 Tesla. The CMS structure can be seen in figure 1.

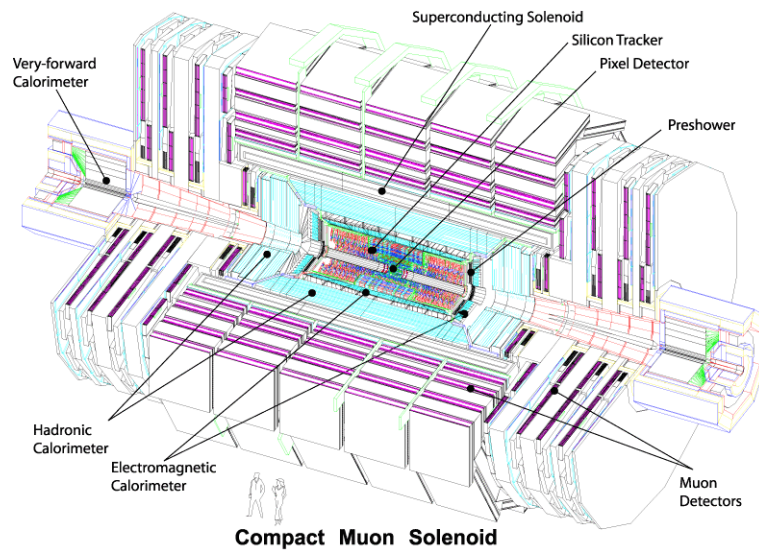


Figure 1: The structure of CMS experiment station. [2]

### 1.2.1 The Pixel Detector

The pixel detector is located at the centre of the CMS detector. It is used to track the charged particles near the interaction region and to provide important pattern recognition aide to the silicon tracker. It has also an important role in the offline analysis of data. A single pixel detector consists of an array of  $150 \mu\text{m}^2$  pixels connected to a pixel readout chip with bump bonding. There are about 45 million pixels in total. The sensors consist of these pixel arrays and the data from the sensors is sent to DAQ via optical fibres.

### 1.2.2 The Silicon Tracker

The outer parts of the tracker form the silicon tracker. There are about 9.6 million  $p^+$  strips implanted on n-type bulk sensors and each strip is a channel. The silicon strip tracker consists of an inner barrel (TIB), which is formed of four cylindrical layers enclosed by three disks (TID) on the both sides. The outer barrel made of six cylindrical layers surrounds the inner barrel and the end-caps are made of nine disks.

### 1.2.3 Electromagnetic and Hadronic Calorimeters

The calorimeters will stop electrons, protons and hadrons and allows their energy to be measured. The electromagnetic calorimeter (ECAL) measures the energies of electrons and photons, as these particles interact electromagnetically. The hadronic calorimeter (HCAL) can measure the energy of hadrons, which interact through the strong interaction.

### 1.2.4 The Muon Detection System

The muons are identified and triggered by the muon system, which uses three different technologies. Drift tubes are used in the barrel region, where the magnetic field is not so intense (maximum about 0.8 Tesla) and the expected particle rates will be relatively low ( $< 10 \text{ Hz/cm}^2$ ). In the endcaps the magnetic field strength can be as high as 3 Tesla and the particle rates are expected to reach  $1 \text{ kHz/cm}^2$ . The technology suitable for the endcap region is Cathode Strip Chamber (CSC), which provides good position resolution and trigger efficiency. [3]

The third type of muon detectors is the Resistive Plate Chamber (RPC) that offers good timing resolution, which enables unambiguous bunch-crossing identification, good rate capability (several  $\text{kHz/cm}^2$ ) and relatively simple design and low cost. These detectors are used both in the barrel and in the endcap region. The basic structure of an RPC

detector can be seen in figure 2. It features a simple single-gap counter, which includes a single gas gap delimited by Bakelite resistive electrodes.

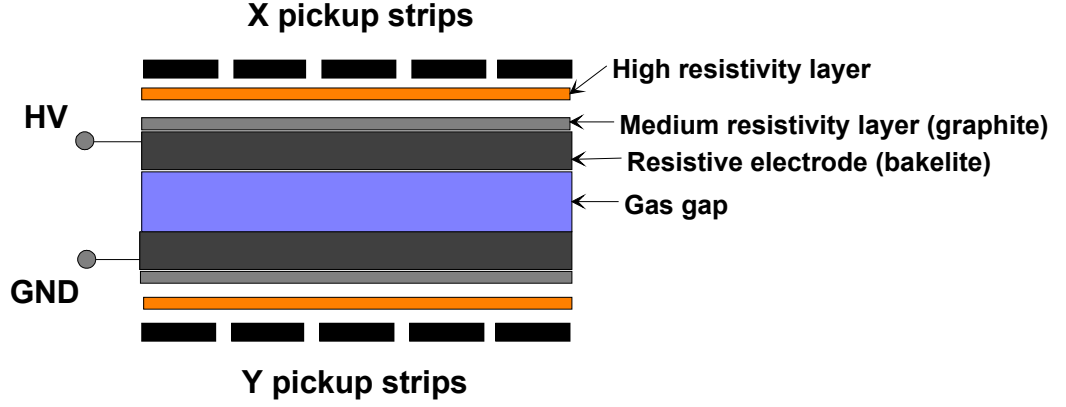


Figure 2: Basic structure of a single-gap RPC detector.

The resistive electrodes are connected to a high voltage generator in order to generate an electric field between the electrodes. The intensity of the field is about 5 kV/mm. The electrodes are coated with a thin layer of graphite so that the high voltage distribution would be as uniform as possible. The gas mixture can include for example tetrafluoroethane ( $C_2H_2F_4$ ), isobutene (iso- $C_4H_{10}$ ) and sulphur hexafluoride ( $SF_6$ ). The avalanche produced in gas gap induces signal on pickup strips placed on both sides of the detector. Besides avalanche mode the detector can also operate in streamer mode, which means that the electrical field is intense enough to initiate a spark breakdown. This phenomenon is not very desirable in detectors equipped with high gain amplifiers and low threshold discriminators because the streamer signals are about 100 times higher than avalanche signals and they can increase the detector dead time. The sulphur hexafluoride in the gas mixture can reduce the number of undesirable streamer events. [4], [5]

### 1.3 Purpose of the work

The purpose of the work is to implement a test set-up to test the communications between the Link Board Box (LBx), Front-End electronics and the Distributed Control System (DCS), which includes a Front-End Controller (FEC) and a PC running Scientific Linux with all the necessary drivers and libraries. The Electronics Design

Centre in Lappeenranta University of Technology has been re-routing the Control Board (CB) and Link Board (LB) schematics as the FPGAs were upgraded from Spartan II to Spartan III. The functionality of the boards with the main parts of the RPC link system can be tested with the set-up. Arja Korpela from Lappeenranta University of Technology has designed a link board tester to ensure the mechanical functionality of the boards. Ahti Karjalainen, Ville Vehmaa and Vesa Väisänen coded the software for the tester. The functionality of the tester is described in chapter 3.6. Figure 3 shows the main principle of the setup with additional Splitter, Trigger and Readout components that are in the link system but not in the set-up.

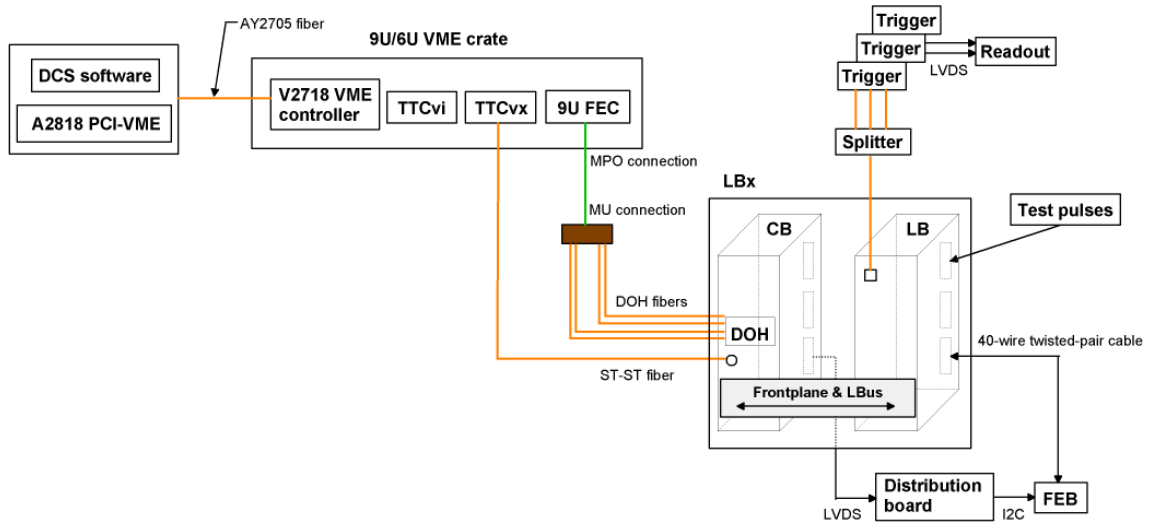


Figure 3: Block diagram of RPC link system.

## 2. Parts of the setup

The purposes of different components are described briefly in this chapter.

### 2.1 Front-end electronics

The RPC front-end electronics consists of Front-End Boards (FEB) handling 16 channels each. Up to six front-end boards are connected via LVDS twisted-pair cables to a link board, which contains synchronization and data compression functions and the optical link transmitter. The data from the link boards is distributed to counting room via optical splitter, which copies the information coming from one MLB to several

trigger boards. The detector chambers are divided in 12 sectors and each sector covers  $30^\circ$ . A single muon can travel through many sections and therefore the data needs to be distributed to several locations. In the counting room the data from each fiber is split between several Trigger Boards (TB) and Readout Boards (RB). [3]

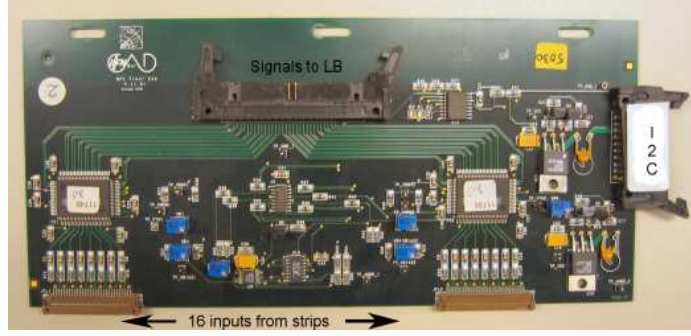


Figure 4: RPC front-end board.

The trigger boards utilize a Pattern Comparator (PAC) algorithm when looking for muon candidates. The chamber hits are compared with predefined muon track patterns obtained from simulations. The selected muon candidates are also sorted and “ghost-busted” and finally eight muon candidates with highest momentum are sent to the global muon trigger. Ghost-busting means that so called ghost muon tracks are removed in as early stage as possible so that they do not interfere the readout of real muon tracks. The ghost tracks are resulting from the RPC detector geometry and 3-out-of-4 logic in the readout, which allows one missing hit in one of the RPC layers. [6]

The control board sends remote control commands to RPC via  $I^2C$  bus. One  $I^2C$  line can control up to eight barrel front-end boards or four endcap front end boards. The  $I^2C$  bus is a local bus running at 100 kbit/s in standard mode. It consists of two serial lines: SCL (uni-directional) and SDA (bi-directional). It cannot drive long lines (few meters max) because it suffers of high capacitive loads. To overcome this limitation, on the control board FPGA the SDA line is split into two uni-directional lines and SCL, SDA\_IN, SDA\_OUT are sent to RPC in LVDS on twisted-pair cable. On the RPC distribution board they are reconverted into standard  $I^2C$  and sent to front-end boards. [7]



Figure 5: RPC distribution board.

## 2.2 The Link Board box

The Link Board Box (LBx) is a crate, which houses up to 18 link boards and 2 control boards. The link board boxes will be placed on the balconies around the detector. The link board boxes gather data from the front-end boards and control them via I<sup>2</sup>C bus. In the RE1/1 sector smaller link board boxes are used and they will be placed in the nose of the endcap. The planned placement of the cards in the link board box can be seen in figure 6.

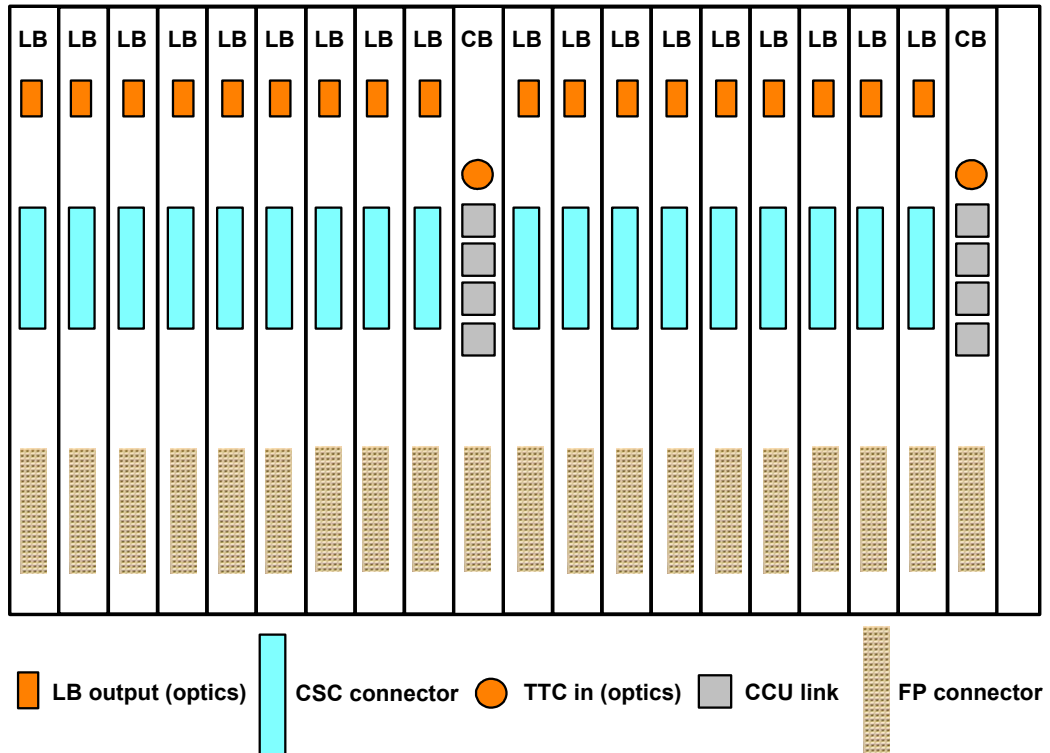


Figure 6: Front view of the link board box.



There are two separate front planes on the link board box. Each front plane connects one control board and nine link boards together. The front planes are attached to the front plane connectors and the required insertion force is obtained by using screws. This mechanism is still under development. The front plane is for the LBus and for the distribution of TTC and LV. The link board optical output uses 850 nm VCSEL (Vertical Cavity Surface-Emitting Laser) with LC connectors and the TTC input is an optical photoreceiver with ST-connector. The Command and Control Unit (CCU) token ring inputs and outputs are standard RJ-45 connectors.

The data and control connections for the front-end electronics and the power supply connectors are on the backplane of the link board box. The backplane structure is shown in figure 7.

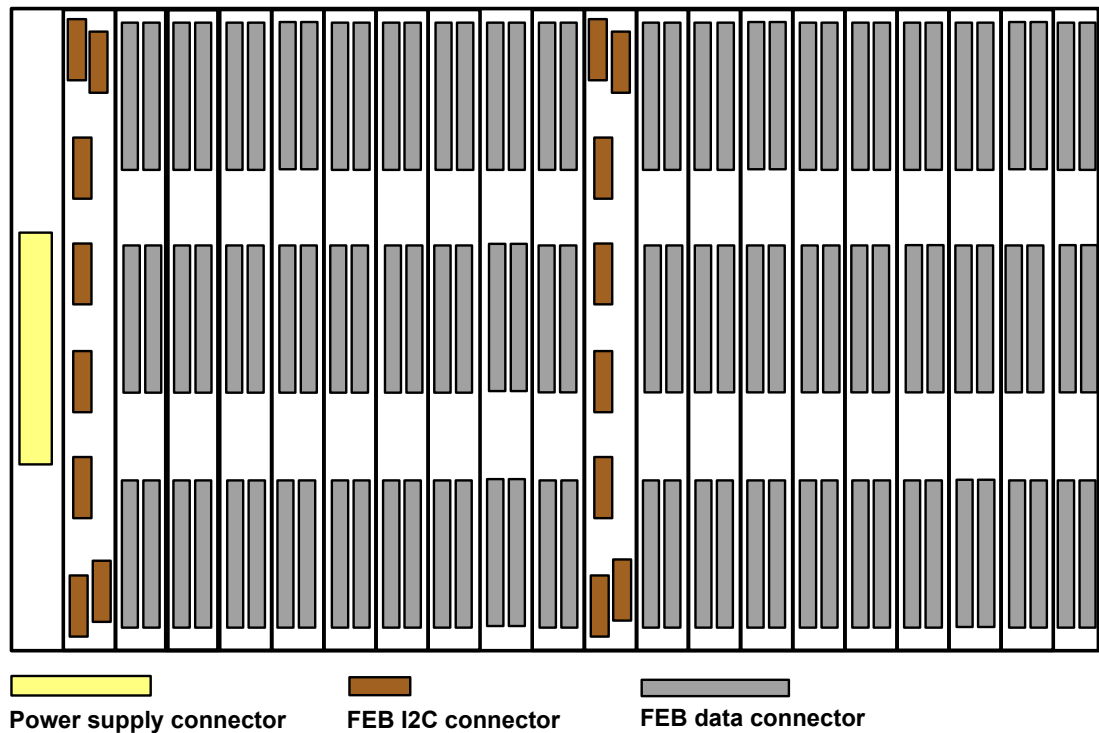


Figure 7: Back view of the link board box.

### 2.2.1 The Link Board

The link board receives data from up to six front-end boards via LVDS twisted pair cables, processes the data and sends it forward through optical fiber.

There are two types of link boards:

- Master Link Board (MLB). The master link board compresses its own data acquired from the front end boards, merges the data with partially compressed data from two neighboring Slave Link Boards (SLB) and transmits it to the counting room through an optical fiber.
- Slave link board. The slave link boards send partially compressed data to master link boards through LVDS twisted-pair cable. The usage of the slave link boards depends on the occupancy of the detector.



Figure 8: The link board v. 1.0 prototype.

There are two Xilinx XC3S1000 FPGAs on the link board: the Link Board Controller (LBC) and the Syncoder, which performs data multiplexing and synchronizes the pulses coming from the front-end boards with LHC clock. The purpose of the link board controller is described the next chapter.

### 2.2.2 The Control Board

The control board can serve up to nine link boards and they are connected together by a local bus (LBus), which is an asynchronous bus with 16 address lines and 16 data lines. The main components of the control board are the Communication and Control Unit (CCU25) chip, which provides the communication with the DCS and controls the local interfaces: I<sup>2</sup>C and LBus, the TTCrx chip that receives and decodes timing and trigger data and the Control Board Programmable Controller (CBPC), which provides an interface between the CCU25 and the LBus. This interface includes special logic to make the communication between the 16-bit data bus of LBus and the 8-bit data bus of the CCU25 possible. The CCU25 and TTCrx are designed in radiation-hard technology in order to prevent single-event upsets that causes data corruption [3].

There are two controllers on control board: Control Board Initialization Controller (CBIC) and Control Board Programmable Controller (CBPC). The CBIC core is implemented in radiation-hard antifuse FPGA and the CBPC is implemented in the reprogrammable Xilinx FPGA.

The CBIC reads the encoded configuration data from control boards' Flash memory and uses the data to configure the CBPC and the Link Board Controllers (LBC) on the link boards. The programming data is transferred on the front plane LBus. If the configuration is successful, the CBIC activates the CBPC and enters the sleep mode, waiting for the next reconfiguration request. If the configuration is not successful (meaning that the Flash contents is not valid) the CBIC enters the emergency mode, in which the CBPC and the LBC are configured with the configuration data sent directly via CCU25. A state machine in the VHDL code handles the different modes. The CCU25 alarm signal is used in this case to notify the managing computer in the DCS, that re-configuration intervention is required. The Front-End Controller (FEC) is the link between the Control Room and the front-end CCU chips.

After the CBPC and LBC chips are configured and activated, the link board controllers are configuring other FPGAs located on the link boards, using the configuration data stored in the link boards' Flash memories. If this configuration is not successful, the

LBC sends an interrupt (which triggers another CCU25 alarm signal) and waits, until the Flash memory content gets refreshed. [8]



Figure 8: The control board v. 1.0 prototype.

## 2.3 Timing, Trigger and Control system

The Timing, Trigger and Control system (TTC) is to distribute the 40,079 MHz LHC clock, 11,246 kHz LHC orbit and individually-addressed control signals to electronics controllers with the appropriate phase relative to the LHC bunch structure, taking account of the different delays due to particle time-of-flight and signal propagation. The overall TTC system architecture can be seen in figure 9 on the next page.



encoded with bi-phase mark encoding by using TTCex modules. The TTCex encoder/transmitter has 10 laser outputs with output power of 0 dBm, which equals to 1 mW. The timing signals from the SR4 to the control room are distributed with existing 9.5 km phase-stabilized optical fibers left over from Large Electron Positron (LEP) collider project. The fibers follow an old railway line before going across countryside to the Preveessin Control Room (PCR). [9]

The first momentous event in Preveessin Control Room was seen in 1976 when the first circulating 400 GeV proton beam was achieved in Super Proton Synchrotron (SPS). The PCR has also been monitoring the events of the LEP since the first electron-positron collisions that took place in 1989. The operations of the PCR will move to new location, when the new combined control center for all the accelerators will be ready. [10]

In the PCR the encoded TTC signal is received with LHCrx module. The clock signals are then synchronized again, encoded with 160 MBaud bi-phase mark encoding, which time-division multiplexes two channels using a balanced DC-free code. The encoded and modulated signal is broadcasted to LHC and other destinations with high-power laser drivers which outputs are fanned out with a 1:32 optical tree coupler. The fiber type used between the PCR and the experiment areas is 9/125  $\mu\text{m}$  singlemode fiber.

The high-power laser driven singlemode fibers coming from the Preveessin Control Room (for redundancy there are two fibers to distribute the same signal) are connected to standard patchboard near the TTCmi machine interface at each experiment. The signal level from the PCR is adjusted to the optimum level of about  $-20$  dBm by a local attenuator. From the optical receiver the TTC signal is distributed to LHCrx module in the TTCmi with special FC/PC – ST/PC patchcord.

The LHCrx module includes a TTCrx chip, which is a custom ASIC designed by the CERN EP Microelectronics group. The circuit receives the bi-phase mark encoded bit-stream via a PIN-photodiode and recovers the LHC clock and 80 Mbit/s serial data from the stream. The TTCrx chip has a watchdog circuit to relock the PLL automatically in case of an optical signal interruption. The TTCrx offers a possibility for clock phase adjustment to compensate differences in particle time-of-flights and optical fiber

lengths. This phase adjustment can be done in scale of local experiments. The LHCrx module has circuits to correct phase differences in the orbit signal received from the PCR at different points of the LHC ring. The orbit phase can be adjusted in 3564 steps of the 25 ns bunch crossing interval. [9]

The TTCmi can be equipped with additional modules in order to meet the TTC requirements of each experiment site. The TTCcf module is used to distribute the orbit clock to TTCvi cards in different trigger partitions. The clock coming from the TTCrx chip has an rms jitter of about 80 ps, which is a bit too high for the primary timing reference for an LHC experiment. The TTCmi module reduces the clock jitter to about 7 ps with a narrow loop bandwidth PLL, which has a low-noise 160.316 MHz Voltage Controlled Crystal Oscillator (VCXO). The TTC signals can be distributed from TTCmi with either laser transmitters (TTCmx) or with led transmitters (LED Optical Tx). The outputs of the laser driver can be fanned out to several destinations by using a 1:32 TTCox optical tree coupler, so the module can broadcast to a total of 128 destinations. There is also a possibility to use external TTCtx laser transmitter placed in a VME crate. The led outputs cannot be fanned out because the led driver output level is enough to drive only one receiver. [11]

Because of economical reasons the fibers used for local TTC distribution at the LHC experiments are multimode 50/125  $\mu\text{m}$  fibers. The front-end electronics that utilize the TTC information for data synchronization or control have a photodiode receiver with an amplifier to receive the optical encoded signal and the TTCrx chip to extract the clock and control data from the stream. The TTCrx chip can be configured with I<sup>2</sup>C and JTAG and it has correction algorithms for single-event upsets. The operation of these algorithms can be monitored via the I<sup>2</sup>C.

### 2.3.1 The VMEbus

The TTC system control is done via VMEbus (IEEE-1014-1987) and the TTCvi module. The VMEbus arrived in 1981 and it was developed for Motorola 68000 based equipment for industrial automation applications. Some advantages of the VMEbus are its mechanical reliability, flexibility and openness, which mean that there are no proprietary rights assigned to it, so anyone can make VMEbus products without royalty fees or licenses. Nowadays the VMEbus is used in a wide variety of applications, such as industrial controls, military command and control systems, aerospace avionics and control systems, transportation controls, telecom applications, various simulations systems and in medical applications.

The original VMEbus specification has the following features [13]:

- Master/slave architecture
- Asynchronous bus
- Variable speed handshaking protocol
- Non-multiplexed bus
- Addressing range from A16 to A32
- Data width from D8 to D32
- Bus bandwidth up to 40 Mbytes/second
- Multiprocessing capability (1-21 processors)
- Interrupt capability
- Up to 21 card slots can be used in a single backplane

There have been three major improvements to the original standard. In 1994 the VME64 was introduced. The addressing and data capabilities were extended to 64-bit and the bus bandwidth was doubled to 80 Mbytes/second. Other improvements include [13]:

- Lower noise connector system
- Cycle retry capability
- Bus LOCK cycles
- First slot detector



- Automatic “plug-and-play” features
- Configuration ROM/CSR capability
- Re-definition of SERCLK and SERDAT pins, also called as SERA and SERB. These pins are used for an optional serial bus such as AUTOBAHN (IEEE 1394) or VMSbus. Under VME64 these pins can be used for any user defined serial bus.

In 1997 the VME64x extension added new capabilities to the VME64 such as [13]:

- A new 160 pin connector family, which is backward compatible with the original 96 pin connectors
- A 95 pin P0/J0 connector between P1/J1 and P2/J2 to meet higher I/O demands in some applications
- +3.3 V power supply pins (located on the connector edges)
- More +5 VDC power supply pins
- Geographical addressing (base address depends on the slot number)
- Bus bandwidth increased up to 160 Mbytes/second
- 141 more user-defined I/O pins
- Rear plug-in units (transition modules)
- Live-insertion/hot-swap capability, no need to power down the whole system in order to add or remove one module
- Injector/ejector locking handles
- EMC front panels
- ESD features

New VME cards are compatible with older backplanes as long as they do not use the +3.3 VDC power supply, as the older backplanes do not have pins for this. Newer backplanes can also provide automatic daisy-chaining while older backplanes require separate jumpers for this.

The PCI-VME interface is carried out by CAEN A2818 controller, which has an 850 nm Vertical Cavity Surface Emitting Laser. The card supports both 3.3 V and 5 V voltages on the PCI bus. Modern motherboards support both of these voltages, but 3.3 V

is more common and the A2818 transmitter can provide more current with the lower voltage. The selection between these two modes is done with a jumper on the controller card.



Figure 10: CAEN A2818 controller.

The A2818 is connected to CAEN V2718 VME controller using AY2705 optical fiber with a duplex connector on A2818 side and two simplex connectors on V2718 side.

The V2718 is a 6U size VME master module. It supports A16, A24 and A32 addressing as well as CR/CSR, LCK, ADO and ADOH cycles. Supported data cycles are D08, D16, D32 for R/W and RMW, D16 and D32 for BLT and D64 for MBLT. There is a DIP switch on the board that needs to be set for desired settings. In this case the switches are:

PROG\_0: ON (system controller enabled, regardless of the 1<sup>st</sup> slot detection)

PROG\_1: OFF (position opposite to PROG\_0)

PROG\_2: OFF (the master initiates the VME cycles with waiting the Bus Grant from the arbiter)

PROG\_3: OFF (the board responds only to A24 cycles)



Figure 11: CAEN V2718 VME controller.

### 2.3.1 TTCvi

The TTCvi is a TTC interface to VMEbus. The card does not have an interrupter nor a bus requester, so the VME backplane daisy-chain jumper positions are irrelevant. Supported addressing modes are A24 and A16 and for data transfers the card supports D32 and D16. TTCvi uses +5 V supply and an on-board DC-DC converter supply the necessary  $-5$  V for the NIM and ECL logic. Current consumption is about 4 A at 5 V.



Figure 12: The TTCvi module.

Some relevant front panel functions are described on the next page. The TTCvi front panel can be seen in appendix 1.

Trigger inputs (**L1A IN<0..3>**): Normally the trigger input to the module is the Level-1 Accept (L1A) signal provided by the Central Trigger Processor. Other trigger sources are also available for test purposes and they are programmable selectable. Triggers can be generated manually via VME access by writing a word in address-offset 0x86. An internal trigger generator is also available and can be configured through the CSR1 register bits <14..12>. The internal trigger generator generates a L1A signal and the number of L1A per unit of time follows a Poisson distribution with a mean rate programmable from about 1 Hz to 100 kHz. Triggers can be monitored via front panel **TRIGGER OUT<0..1>** NIM outputs. [14]

Clock input (**CLOCK IN bc/ecl**): The clock input is for 40,079 MHz LHC clock. The internal oscillator of TTCvi can generate this clock, if the external clock is missing. [14]

Orbit input (**ORBIT in/ecl**): The orbit signal frequency is 11,246 kHz which is received from the TTCmi and distributed to the TTCvi and other components for the generation of signals which require to be synchronized to the LHC orbit. The orbit signal can be generated internally for testing purposes. Setting bit 03 in CSR1 register to '0' makes the selection between external and internal orbit signal. The selected signal is available via front panel NIM output called **ORBIT out/nim**. If the external orbit signal is selected and its connector is connected to the orbit input, the front panel indicator ORBIT is lit. The indicator is also active with the internal orbit signal. [14]

The A channel signal is used to transmit the L1A signal and B channel is for framed and formatted commands and data. The data commands can be either:

- Short-format synchronous or asynchronous broadcast command/data cycles. If synchronous, the timing of these cycles relative to the LHC orbit is controlled precisely. These broadcast commands are used to distribute messages to all TTC destinations in the system and all the TTC receivers execute them when they detect the command.
- Long-format asynchronous individually addressed or broadcast command/data cycles. These commands have two modes of operation. In the first mode the commands are intended for the TTC receivers themselves and in the second

mode the commands are intended for external electronics. This information can include the transmission of parameters, test data, calibration data and non time-critical commands, such as channel masks, to the front-end electronics. The frame format of data transmission is described in figure 13. [15]

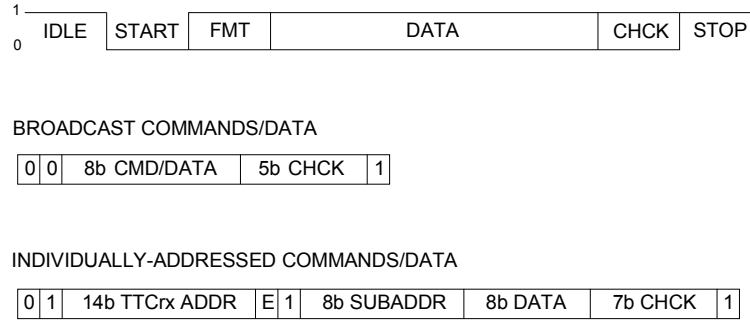


Figure 13: TTC data transmission frame format. [15]

The data transmission starts with logical “0” and stops with logical “1”. There is a header bit (FMT) in the beginning of each frame. The header “0” is used when sending broadcast commands and for individually addressed commands the header bit must be “1”. The address selection bit (E) is used to indicate if the operation is meant to be internally executed (logical “0”) or if the command/data should be made available for external electronics (logical “1”). There is also error correction for protecting the data from corruption. The coding scheme is a standard Hamming code with the capability of double error detection and single bit error correction. The internal addressing space is allocated as seen in table 1. [15]

Table 1: TTCrx internal addressing space.

SUBADDR <7:0>	Register/Command
00000000	Fine Delay Register 1 <7:0>
00000001	Fine Delay Register 2 <7:0>
00000010	Coarse Delay Register <7:0>
00000011	Control Register <7:0>
00000100	Execute ERDUMP command
00000101	Execute CRDUMP command
00000110	Execute RESET command

### 2.3.2 TTCvx

In our test set-up the TTCmi minirate is replaced with TTCvx, which is a low-power VME 6U –size led transmitter module. The function of the TTCvx is to multiplex and encode the A and B channels generated by the TTC-VME bus interface module TTCvi. The channels are multiplexed by time division multiplexing and encoded with bi-phase mark encoding. The encoded TTC is then distributed to the link board box via optical multimode fiber. The TTCvx has no VME capabilities and it uses only the power supply buses on the backplane. The picture of the TTCvx front panel is in appendix 2.

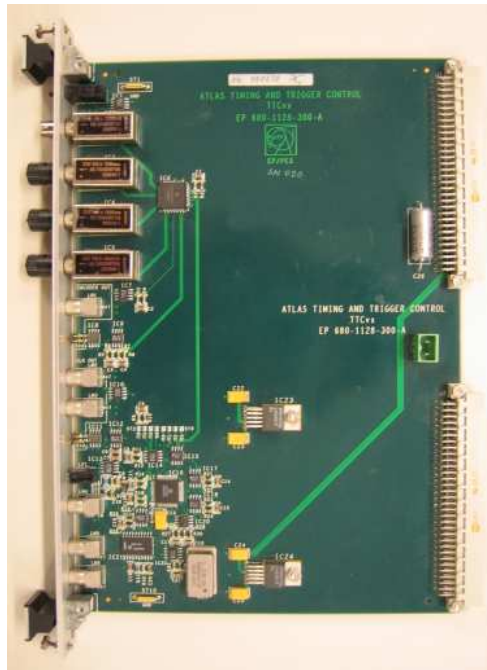


Figure 14: The TTCvx module.

The TTCvx front panel functions are the following:

**Channel A/B In:** Inputs for the A and B channels from TTCvi. These inputs are DC-coupled and internally terminated with 50  $\Omega$ . Both inputs are biased to ECL “0” when not connected.

**Clock In:** Input for external ECL clock. The front panel led indicates if external clock is present. This input is AC-coupled and internally terminated with 50  $\Omega$ . There is an 80

MHz quartz ECL oscillator on board and the clock is generated internally if the external clock source is not connected.

**Clock Out:** Two AC-coupled ECL level outputs and two LVDS clock outputs used for synchronization of external equipment, in this case the TTCvi (ECL). These outputs carry the basic frequency of the PLL circuit.

**Encoder Out:** One AC-coupled ECL level output and two LVDS outputs carrying encoded TTC signal generated from A and B channels. The LVDS signal can be used for connecting to the TTCrx end in systems without optical transmission, as the TTCrx chip has LVDS inputs for the TTC signal.

**Fiber Optics Out:** These outputs are four fiber optics transmitters of LED type (1330 nm) with output power of  $-14$  dBm, which equals to  $31.81 \mu\text{W}$ . These outputs are connected to receiving TTCrx chips via optical fibers. The fibers used with the TTCvx in the set-up are multimode 50/125  $\mu\text{m}$  fibers with ST-connectors at both ends.

**PLL Reset:** Push button to reset the PLL circuit if phase locking is lost.

## 2.4 The Detector Control System

### 2.4.1 DCS overview

The environment in the detector is very hostile for electronic devices. The front-end electronics must be able to survive the radiation and recover from single event upsets while monitoring many environment parameters such as bias currents, temperatures and voltages. The programmable chips together with radiation-hard technology make it possible to monitor the condition of the readout system and if data corruption is seen the DCS system must be able to reconfigure the front-end electronics as quickly as possible. The tasks of the detector control system include communication with the managing host (XDAQ software and databases, which contains configuration information), transmission of the control and diagnostic data to and from the front-end electronics,

transmission of the control and diagnostic data to and from the link boards and refreshing of the configuration of the programmable FPGA chips.

### 2.4.2 DCS hardware

The hardware is based on the Front-End Controller (FEC) board, which is the interface between the CCU token rings and the control software. The card is not the final version, but a V2 prototype. The front-end controller is a 9U card, which is located in the CMS control room VME crate. In our test set-up the card is housed in the same crate with 6U size TTCvi and TTCvx. This requires slight modifications to 9U crate so that the smaller cards will stay in place. The FEC-CCS complies with the VME64x standard and uses the +3.3 V pins from the backplane as well as geographical addressing. The current consumption of the controller is about 7 A at 3.3 V and 1 A at 5 V.

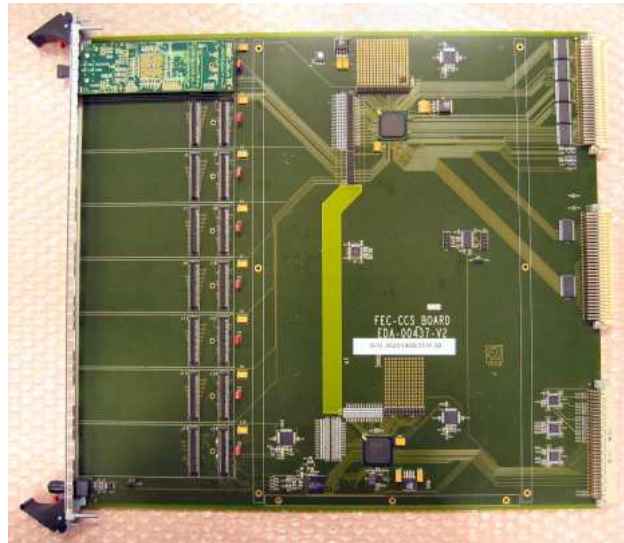


Figure 15: 9U FEC-CCS board.

The control ring is based on a token ring topology where the CCU chips are connected together with the optical mezzanine board on the FEC. Each frame to be sent contains the FEC number, CCU address, channel number, transaction number to recover the action which initiates the frame and a command number to tell which command to execute (read, write, read-modify-write). The I<sup>2</sup>C accesses require also an address and the data to write. The CCU is designed to be radiation resistant, but there is a cyclic redundancy checksum and an auto-correction algorithm implemented on the chip in



case of single event upsets. Faulty CCU chips can be bypassed by configuring the inputs and outputs of each CCU and FEC. There are two separate rings, A and B, to provide the redundancy. The CCU chips have different channels to communicate with the front-end chips, such as 16 I<sup>2</sup>C channels, parallel inputs and outputs (PIO) and controllers for these buses, one memory channel to access external devices such as static memories or analog-to-digital converters, one JTAG master controller, one trigger distribution controller and one node controller to be able to report the status of other channels if necessary. [16]

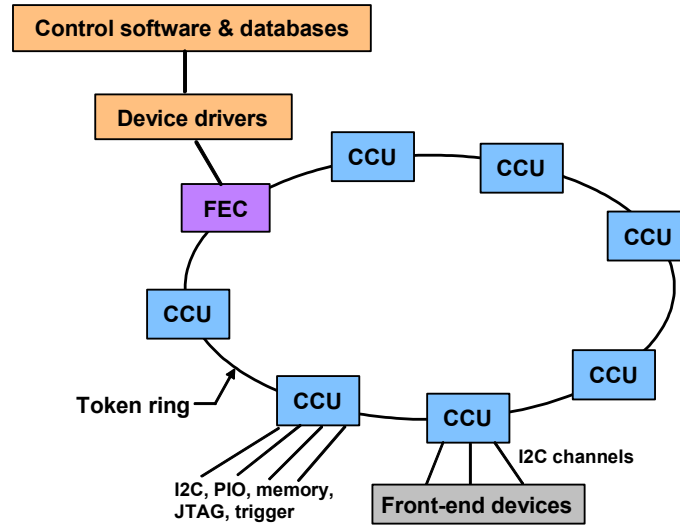


Figure 16: The simplified DCS hardware architecture.

There is room for eight optical mezzanine boards on the FEC. Each optical mezzanine board can be connected to one token ring. In the RPC link system it is done via Digital Optohybrids (DOH) attached to the control boards with 12-way MU/MPO terminated optical fanout. The MPO connection is on FEC side and the MU connection is on the DOH side. The digital optohybrid has four channels: two for clock and data input and two for output and it acts as an interface between the optical and electrical signals. The DOH and the fiber connections can be seen in figure 17.

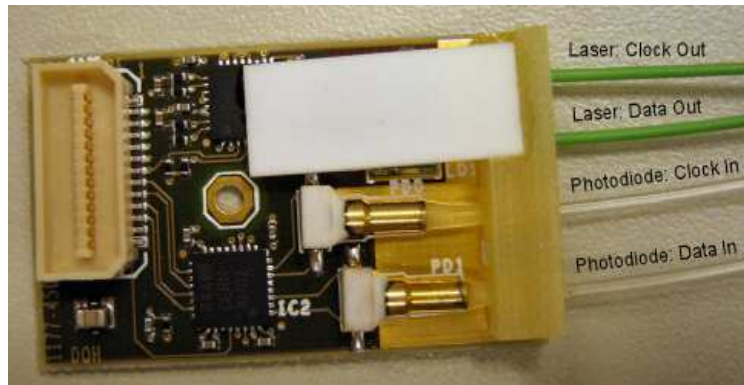


Figure 17: The Digital Optohybrid.

There are four leds on the front panel of each mezzanine board. The PROG leds indicate if the FPGA chips on the FEC have been programmed correctly. Successful programming is indicated by the green led. The LINK leds indicate the status of the data link. If the FEC is connected to a CCU ring or if there is a loop, the green led should be constantly on. The red led is on if there are communication errors or if the link is not connected at all.



Figure 18: Front panel of the optical mezzanine board.

### 2.4.3 DCS software

The functionalities of the DCS software are divided in different processes. The processes of the system can be seen in figure 19. The functionality of the DCS software system is essential, because it has to ensure that the detector is working properly. It has the capability to prevent the detector from damage in case of a hardware failure by re-configuring the hardware or by shutting off different parts in the system if the new parameters cannot be uploaded.

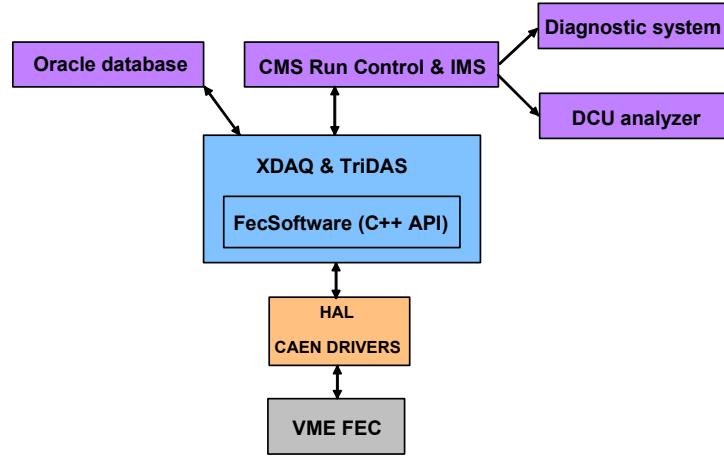


Figure 19: The simplified DCS software architecture.

The platform for the software is Scientific Linux, which is a rebuild of the free Red Hat Enterprise Linux 3 distribution. The control software runs under XDAQ framework, which provides platform independent services, tools for local and remote inter-process communication, configuration and control, as well as technology independent data storage. The FecSoftware program components utilize these services as they communicate with the databases and the Run Control system. TriDAS comes from the words Trigger and Data Acquisition and it covers the hardware and software needed for the control system.

The central part of the system is the FecSoftware, which communicates with the front-end controllers in order to read and write parameters into front-end electronics: the Tracker modules in the Silicon Tracker and the control boards in the RPC. The control board sends the I<sup>2</sup>C data in LVDS to the distribution board, which makes the LVDS to I<sup>2</sup>C conversion and distributes the data to front-end boards. In the RPC front-end boards the configurable parameters are the signal threshold and pulse width. All other components on the board are I<sup>2</sup>C slaves such as analog-to-digital and digital-to-analog converters.

The interface between the hardware and the software is the driver. The Hardware Access Library (HAL) has been developed in order to make hardware changes possible without re-writing the whole software to support the new hardware. With HAL it is possible to change for example the VME controller and only changes needed for the

software is to change a couple of lines from the configuration file (which can be either in XML-format or in ASCII) and the HAL will do the rest. The user program utilizes the `HardwareDevice` class in the HAL when communicating with hardware. The different hardware registers and the functions of their bits are stored in separate address tables as items, so each item has a logical name to describe its functionality, addresses, bitmasks and information that tells which bits can be written or read. There are no visible addresses in the source code; there are only strings to identify the logical items. An example from the `FecAddressTable` used in the set-up shows the mappings for a VME FEC in slot 8 with one CCU ring. The values from the left are the name of the hardware item, the VME address modifier, the width, VME address, bit mask and on the right there are the two bits for read and write (with 1 the operation is permitted and denied with 0).

Table 2: Address table example taken from `FecAddressTable` used in the set-up.

```
*****
* mapping of SLOT8 RING 0
*****
CONTROL0_8_0      09 4 40000000 ffffffff 1 1
CONTROL1_8_0      09 4 40000004 ffffffff 1 1
STATUS0_8_0       09 4 40000008 ffffffff 1 0
STATUS1_8_0       09 4 4000000C ffffffff 1 0
VERSION_SRC_8_0   09 4 40000010 ffffffff 1 0
TRA_FIFO_8_0_R    09 4 40000020 ffffffff 1 0
RET_FIFO_8_0_R    09 4 40000024 ffffffff 1 0
REC_FIFO_8_0_R    09 4 40000028 ffffffff 1 0
TRA_FIFO_8_0_W    09 4 40000020 ffffffff 0 1
RET_FIFO_8_0_W    09 4 40000024 ffffffff 0 1
REC_FIFO_8_0_W    09 4 40000028 ffffffff 0 1
```

The hardware specific device driver is the low-level access to the hardware. In the HAL there are `BusAdapters` for different systems that act as an interface between the hardware access library and the device driver. There are also so called dummy `BusAdapters` to give the programmer a possibility to debug the software without having any real hardware connected. In dummy mode the computer memory is used for read and write commands. [17]

The interval of hardware accesses depends on the conditions in the detector front-end. The `FecSoftware` sends the correct parameters to the front-end electronics and after the transfer the data is read from the devices and compared with the values sent. If there are

differences the register information of a faulty device is stored in the database and the Information Message Service (IMS) is used to send an error report to the run control. The diagnostic system then decides what to do with the faulty devices. The Oracle database includes the configuration parameters for each device in different sectors.

### **3. The test set-up**

In this work there are three types of tests that have been made with the link board box. The link board tester has not yet been tested with the final link boards. The test methods and their results will be introduced in this chapter. The software used with the hardware will also be introduced in conjunction with each test.

#### **3.1 The TTC system and the Front-End Controller**

In CMS the front-end controllers are placed in Wiener VME64x powered crates, manufactured by WIENER Plein&Baus GmbH, together with the CAEN VME controllers. In our test set-up the 9U front-end controller is located in the same crate with the 6U VME controller and with the TTCvi and the TTCvx. The crate is 9U, but there is a divider module installed for the smaller cards. The front-end controller requires the VME64x backplane so there is a need for power supply that can provide also the 3.3 V together with 5 V, +12 V and -12 V. As it was pretty difficult to find such power supply in time the 3.3 V was generated with the Splitter board. It has a power supply, which takes 5 V from the backplane and converts it to 3.3 V. This power supply can provide about 10 A current for the 3.3 V lines in the VME backplane.

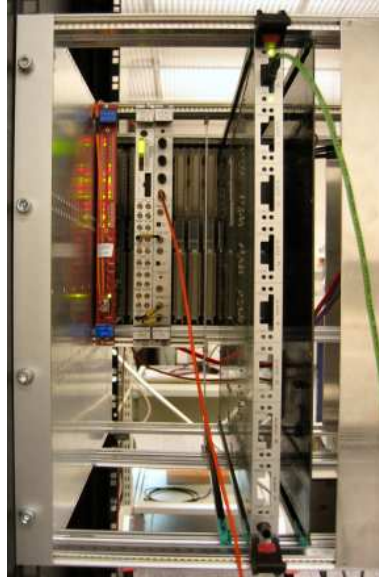


Figure 20: The TTC/FEC crate used in the set-up.

The basic functionalities of the TTCvi and FEC were tested. The software used for the manual VME access was *CAENVME Demo*, which can be obtained from the CAEN website. It provides all the basic functions needed for accessing the VME devices. The TTCvx has no VME capabilities so it does not react to software commands in any way. The main function of the TTCvx, which is the TTC encoding, can be tested together with the TTCvi.

### 3.1.1 Testing of the TTCvi and TTCvx

The base address of the TTCvi needs to be set before any commands can be given. The address is set with four hexadecimal rotary switches located on the board. In this case all the four switches were set to “D”, which makes the base address DDDD00. The supported addressing mode is A24 and the proper data mode is D16. Supported address modifiers can be found from the TTCvi manual. The address modifier cannot be changed with the VMEDemo. The correct modifier is selected automatically.

When all the interconnecting LEMO cables between the TTCvi and the TTCvx are connected and the crate is powered, the BC-EXT, ORBIT and the L1A-SEL leds should be on. This means that the external LHC and ORBIT clocks are available (generated internally if not connected) and the L1A-SEL leds indicate that L1A triggers are

disabled. When TTCvi A and B channel outputs are in idle mode the A channel is always low (ECL) and the B channel is always high. This should be seen also in the TTCvx encoder output. Figure 21 shows the encoder output (channel 1) in relation to the TTCvx clock out (channel 2). In bi-phase mark encoding logic '1' is represented by a level change in the middle of the bit and with logic '0' there is no such level change. The ECL encoder output is working properly.

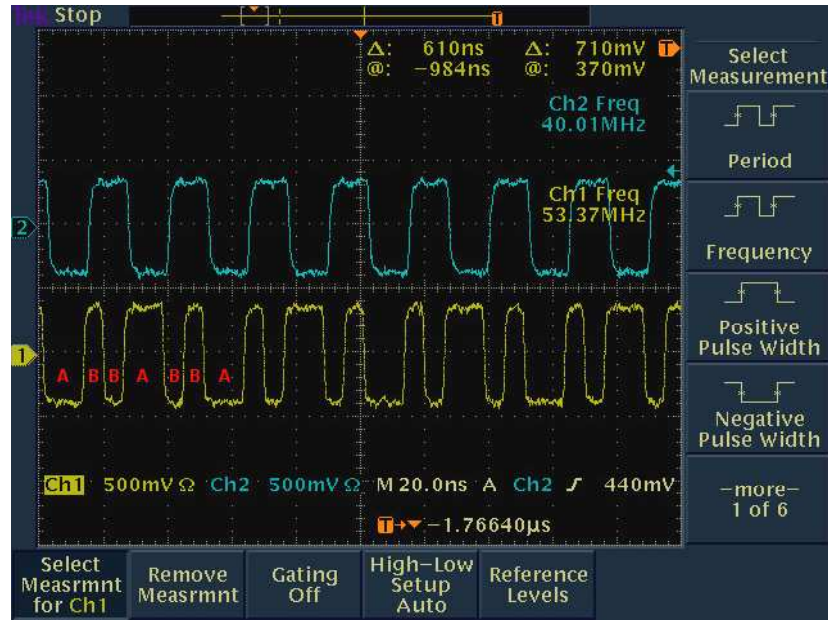


Figure 21: TTCvx ECL encoder output in relation to clock.

The TTCvi A channel carries the L1A trigger pulses. In CMS the trigger pulses are generated at the central trigger processor and then distributed to TTCvi trigger input. The trigger pulses can also be generated internally by changing the L1A trigger select bits in the CSR1 input selection and timing register. The functions of this register can be seen in table 3. [14]

Table 3: TTCvi input selection and timing register CSR1.

Bit	Read/Write	Function	Description
15	R/W	Event/Orbit count selection	'0' = Event count, '1' = Orbit count
14	R/W	Random trigger rate MSB	'7' = 100kHz, '6' = 50k, '5' = 25k
13	R/W	Random trigger rate MSB	'4' = 10k, '3' = 5k, '2' = 1k
12	R/W	Random trigger rate LSB	'1' = 100Hz, '0' = 1Hz
11	R	BC delay MSB	Read BC delay switch value
10	R	BC delay	2 ns/switch step
9	R	BC delay	
8	R	BC delay LSB	
7	R	VME transfer pending	VME request is still pending if '1'
6	W	L1A FIFO reset	Reset with '1'
5	R	L1A FIFO empty	FIFO is empty if '1'
4	R	L1A FIFO full	FIFO is full if '1'
3	R/W	Orbit signal select	'0' = External ORBIT, '1' = Internal ORBIT
2	R/W	L1A trigger select MSB	'5' = Random, '6' = Calibr, '7' = Disabled
1	R/W	L1A trigger select	'3' = L1A<3>, '4' = VME trigger
0	R/W	L1A trigger select LSB	0' = L1A<0>, '1' = L1A<1>, '2' = L1A<2>

To test the TTCvx encoder output response to random 1 kHz triggers we have to write 0010000000001101 (0x200D) to CSR1 register offset 0x80. The offset is summed to the base address, so the address where to write is 0xDDDD80. The appearance of the VME software user interface after a successful write access can be seen on the next page.



## CAEN VME Manual Controller

```

R - READ
W - WRITE
B - BLOCK TRANSFER READ
T - BLOCK TRANSFER WRITE
I - CHECK INTERRUPT
1 - ADDRESS                [00DDDD80]
2 - BASE ADDRESS           [00DDDD00]
3 - DATA FORMAT           [D16]
4 - ADDRESSING MODE        [A24]
5 - BLOCK TRANSFER SIZE    [256]
6 - AUTO INCREMENT ADDRESS [OFF]
7 - NUMBER OF CYCLES       [1]
8 - VIEW BLT DATA
F - FRONT PANEL I/O
Q - QUIT MANUAL CONTROLLER

```

```

Cycle(s) completed normally
Write Data [hex] : 200D

```

After successful VME cycle the TTCvi front panel led for A channel activity should flash on every L1A trigger signal. The trigger pulses are active high ECL pulses on the A channel with pulse width of 25 ns. The TTCvx encoder output response to the trigger pulses can be seen in figure 22.

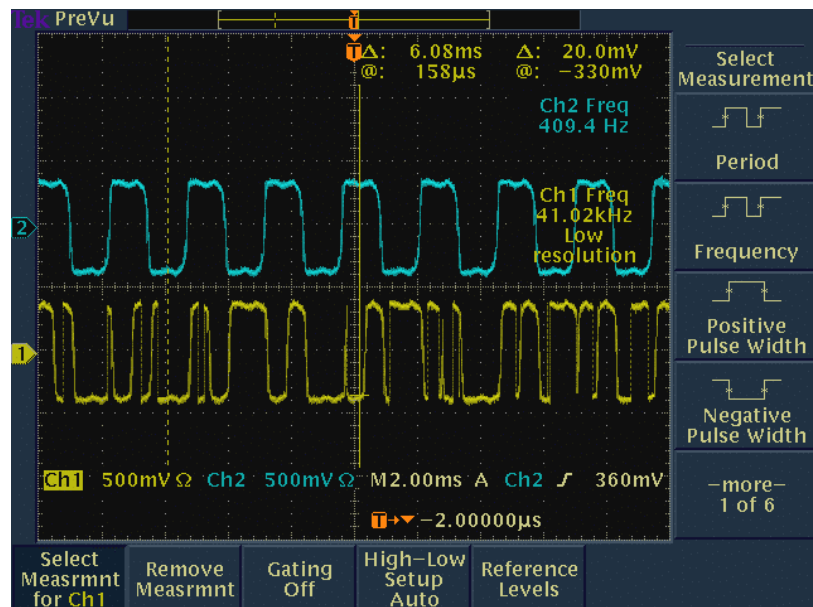


Figure 22: TTCvx ECL encoder output with random 1 kHz L1A triggers.

The L1A triggers can be seen in encoder output. The A channel is no longer always low as in idle mode. It goes high in random intervals when the trigger pulse is initiated and the coding scheme represents these high states by level change in the middle of the bit. The tests were done using 2 ns LEMO cables. The frequency of the encoder output is 80 MHz, so with longer cables the signal gets heavily distorted. The optical outputs of the TTCvx were tested using Tektronix TDS7404 oscilloscope with OE-2 Wavecrest optical module attached. The fiber used was a 2 meter 50/125  $\mu\text{m}$  multimode fiber. The gain of the OE-2 was set to 20 dB. The optical encoder output with idle TTCvi A and B channels can be seen in figure 23. The encoded signal behaves as expected.

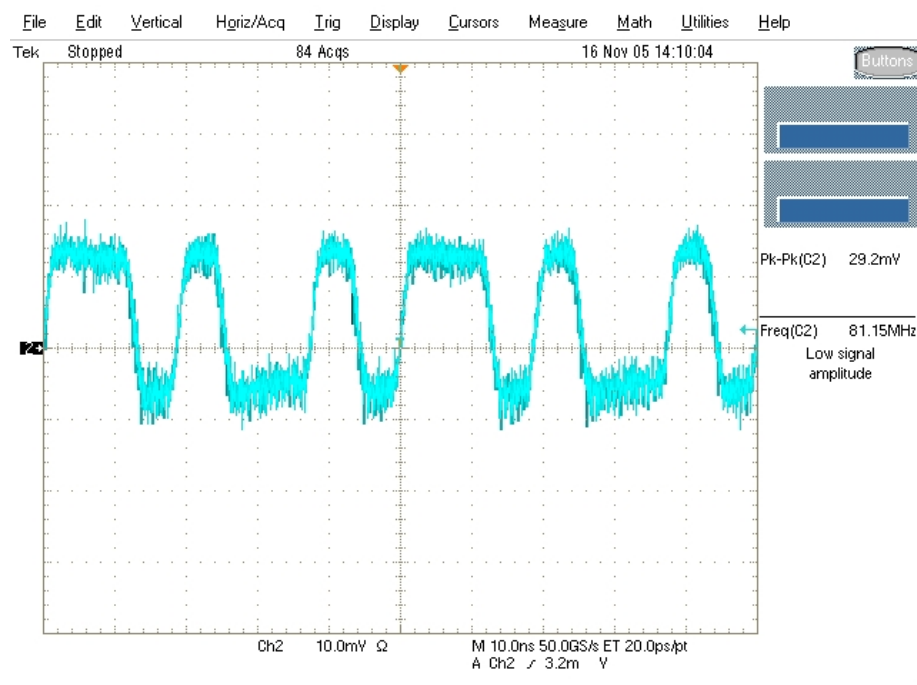


Figure 23: TTCvx optical output with idle A and B channels.

The triggers can be generated also manually by writing for example 0x380C to the CSR1 register and after that 0xFF to offset 0x86 in order to generate a single trigger pulse. If event counting is enabled from the CSR1 (bit 15 set to '0') every L1A trigger increases the value of TTCvi event counter register by one. The number of events can be read from offsets 0x8A (bits 0..15) and 0x88 (bits 16..23). If the event counter needs to be reset it can be done by writing 0xFF to offset 0x8C. The same counter can count also the ORBIT events, if bit 15 in the CSR1 register is set to '1'.

To test the B channel activity with repetitive broadcast command cycles the following initialization routine can be carried out:

- Write 0xFFFF to offset 0x84 for software reset.
- Write 0x6000000 to offset 0xB0 to load FIFO with data. This data should now include broadcast command to reset the bunch and event counter in TTCrx chips. The actual command is 8 bits long and the format is XXXXXX11. The 'X' stands for user-defined value, which does not affect to the reset command. The bunch counter is a 12-bit counter that is incremented by the 40 MHz clock. The event counter is 24 bits wide and it is incremented by L1A triggers.
- Write 0x0000 to offset 0x82 to set the B-Go command to be re-transmitted when the FIFO is empty.
- Write 0x0005 to offset 0x94 to set the Inhibit<0> duration in number of clock cycles. The Inhibit signals are programmable timing signals for sending synchronous commands at user-defined times relative to the LHC orbit. The synchronous command is sent at the end of the Inhibit signal duration. The Inhibit signals have different priority levels. The Inhibit<0> has the highest priority and the commands after this signal are executed first.
- Write 0x000D to offset 0x90 to set B-Go<0> mode. With this data the FIFO status is ignored when starting a cycle, repetitive and synchronous mode is enabled and the front panel B-Go input is disabled. [14]

After these commands there should be data on the B channel and the TTCvi front panel led should be on.

### 3.1.2 Testing of the front-end controller

The functionality of the FEC can be seen roughly from the leds on the optical mezzanine board. If the PROG led is green the FPGA chips on the front-end board are fine. If the CCU ring is disconnected the red LINK led should be constantly on and when the CCU ring is properly closed the green led should be on. The board status register value tells if the front-end controller has been properly initialized during reset. Resetting can be done with *ProgramTest.exe* belonging to the FecSoftware package

developed by Frederic Drouhin. The status register value should be 0x490 with CCU ring connected or 0xC90 without ring. The syntax for the reset is

```
./ProgramTest.exe -vmecaenpci -fec <slot> -reset
```

The test run with CCU ring connected gave the following results:

```
[s29302]/home/ajaja/bin > ./ProgramTest.exe -vmecaenpci -fec 8 -reset

VME FEC will be used with the file
/usr/local/xdaq/TrackerOnline/2005/FecSoftwareV3_0/config/FecAddressTable.dat
Make the configuration for
  - reset all PLX and FECs
Press <Enter> to continue ...

-----
Reset PLXs and FECs
-----
A crate reset is done, not a board or a FEC reset
*****FecRingDevice::fecRingReset
Value of the Status Register 0 of the FEC 8 ring 0: 0xc90
      FIFO receive empty
      FIFO return empty
      FIFO transmit empty
      Link initialise
```

The status register value is 0xC90 with the ring and without the ring the return value is 0x490 so the front-end controller responds to commands and returns the correct data.

### 3.2 Programming of the CBIC

The testing with the link board box was done with version 1.0 prototypes of the control board and link board. The backplane and the frontplane were also prototype models. The picture of the link board box with frontplane, fibers and RJ-45 cables attached is in figure 24.

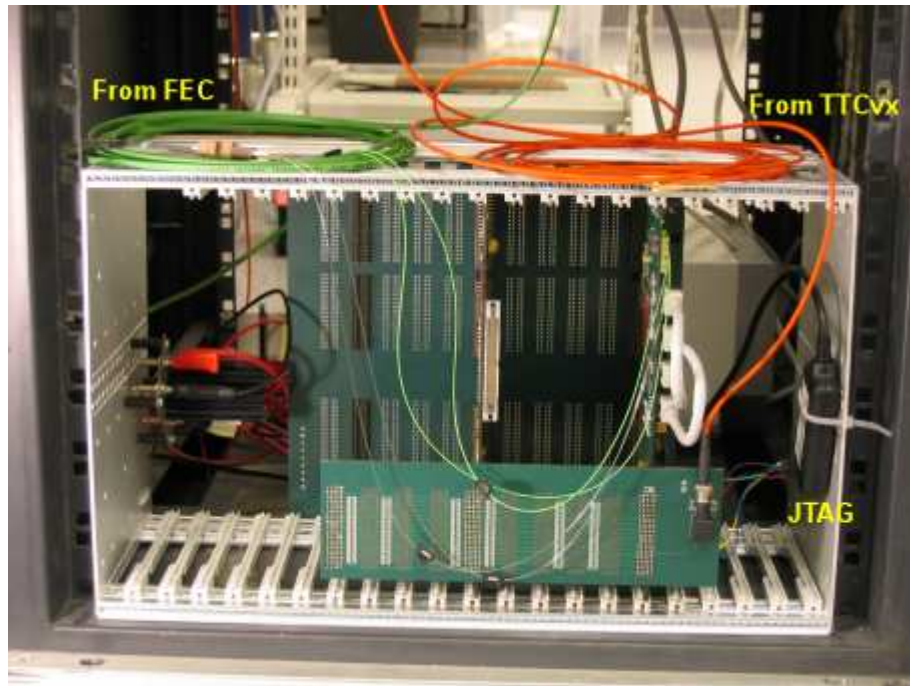


Figure 24: The link board box with version 1.0 prototype CB and LB.

One difference between the version 1.0 prototypes of control and link boards and the final versions (besides the fact that the new boards are bigger) is that the prototypes do not have PROM memory to store the FPGA configurations. Therefore they have to be re-programmed after every power-up. The programming of the CBIC is done through JTAG. This allows us to program the controller with any JTAG compatible programming software. Two different programming software has been tested: *Xilinx ISE* package and a small program called *xilinx-jtag* written by Rene van Leuken.

The *xilinx-jtag* executable takes two parameters: the binary file to be loaded and the parallel port device (usually `/dev/partport0`). A script called *boot\_cb* calls the executable file with proper parameters and the programmer starts to load the CBIC via JTAG. The program reads the FPGA device ID and if the least significant bit of the ID is not '1' the program gives an error, as there is something wrong with the device or in the cable connections. The FPGA codes for the boards are written by Wojciech Zabolotny from Warsaw University.

The programmer output can look like this after successful programming:

```
[s293026] /home/ajaja/bin > . boot_cb
Design Name: cbic.ncd
Device: 2s300eft256
Date: 2004/ 6/16
Time: 23:10:14
Bitstream Length: 1875648 bits
Device ID: 90a20093
Manuf: 49, Part Size: 20, Family Code: 5, Revision: 9

Programming ... *****
Programmed 1875648 bits
```

### 3.3 Programming of the LBC and Syncoder

The principal of CCU activated programming of the link board FPGA chips is described in chapter 2.2.2. There are also JTAG connectors on the link board that allow the FPGAs to be loaded with standard hardware programming. In our set-up the programming of the LBC and the Syncoder is done with *RPCT* software written by Michal Pietrusinski from Warsaw University. The software uses the CCU memory and PIO channels to communicate with the FPGAs. The FecSoftware and its components are the interface between user programs and the CCU channels.

The software package has two main executables for link board programming: *ccu\_server* and *ccu\_client*. The client software takes user commands and communicates with the server program, which prints the responses from the different operations on the screen. The first step is to load all the necessary Linux environment variables. These are included in a script called *env.sh*, which adds the XDAQ libraries to LD\_LIBRARY\_PATH. After loading the variables the CBIC must be programmed as described in the previous chapter before the link board can be accessed. Before starting the server and client programs the front-end controller and CCUs should be reset by using the ProgramTest.exe diagnostic tool provided with the FecSoftware. The reset commands are the following:

```
./ProgramTest.exe -vmecaenpci -fec 8 -reset (for FEC reset)
./ProgramTest.exe -vmecaenpci -fec 8 -filecommand ~/bin/memBusTest.txt
(for CCU reset)
```

The `filecommand` parameter executes a command cycle from a file called `memBusTest.txt`. The file includes data frames for resetting the CCU, enabling the CCU channels and accessing the memory bus.

After reset procedures the `ccu_server` can be launched. The program returns a few values from CCU control and status registers. They are provided by a `FecSoftware` library. The output looks like this:

```
[s293026] /home/ajaja/bin > ./ccu_server
FEC 8 ring 0
CR0 = 0x0
SR0 = 0x3c80
SR1 = 0x8
DONE
```

After the `ccu_server` has started the `ccu_client` can be run. It returns the same register values as the server program. When the client is ready to accept commands there will be a command prompt visible. The first command to give is `scr mscb`. It will run a script which includes the following commands:

```
plbc mlbc.bin
pcbpc fcbpc.bin
wp 1 ef
wp 3 fc
R 7012
R 7013
R 7014
```

The first command of the script loads the link board controller with file `mlbc.bin`. The second command is for loading the CBPC. The `wp` command writes data to the CCU PIO channel and the `R` command reads data from the memory channel. The result of the read operations should be `0xFFEF` after successful programming. The state of the CBPC can be seen from eight green leds on the control board: they should be on after the programming.



Figure 25: The CBPC indicator leds after successful programming.

The link board controller has also indicator leds attached and they should be on after running the programming script. The `ccu_server` program shows hexadecimal data of the programming process while executing these commands.



Figure 26: The LBC indicator leds after successful programming.

After running the first script command the link board Syncoder can be programmed. It is done with command `px 4 rpc_lbstd_syncoder.bin`. Number 4 in the command tells the program in which frontplane slot the link board is connected. The file is again the bit file for the FPGA. The logic in the frontplane numbering is described in figure 27. The control board number is always `-1` and the link board numbering increases to the left.



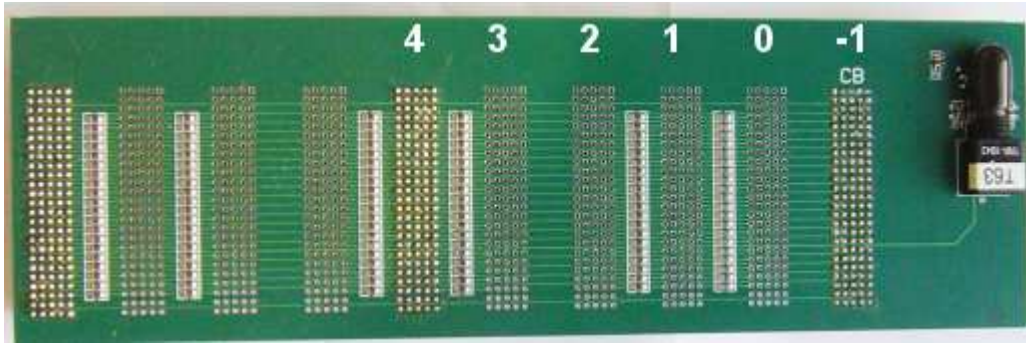


Figure 27: The numbering of the frontplane slots when using the RPCT software.

The software reads a configuration file called *system.xml*, in which the different hardware items and their addresses are described. Every linkbox and most of the chips in the hardware have their unique identifiers. The *ccu\_server* begins to show hexadecimal data of the programming when the command is entered in the client. After this sequence the link board Syncoder should be loaded. With the version 1.0 prototypes the state of the Syncoder can not be seen from leds as there are no leds for this purpose. In the final link board design there are four leds attached to the Syncoder that can be used to indicate that the FPGA has been loaded. The whole set of *ccu\_client* commands needed for programming the chips can be seen below.

```
[s293026] /home/ajaja/bin > ./ccu_client

FEC 8 ring 0
CR0 = 0x0
SR0 = 0x3c80
SR1 = 0x8
>scr mscb
ffef
ffef
ffef
>px 4 rpc_lbstd_syncoder.bin
```

The RPCT software has not yet been documented in public, so the usage of the programs and the meanings of the different commands have been discussed in private with the developer Michal Pietrusinski and his colleagues.

### 3.4 The CCU ring

In the RPC link system the CCU chips are on the control boards housed in the link board boxes. The token ring between the control boards is implemented with RJ-45 cables. In theory there can be 127 CCU chips in one token ring as the addresses go from 1 to 127, but the practical maximum has not yet been tested. As already mentioned in the previous chapters, the front-end controller has a quick led indicator for continuity of the token ring. The data going in ring A input goes to CCU25 and from there to the ring A output and eventually the closed token ring returns back to the front-end controller via the digital optohybrid. The ProgramTest.exe includes a few parameters to scan the token ring for CCU chips. The command to check the device driver, FEC and CCU chips in the token ring is

```
./ProgramTest.exe -vmecaenpci -fec 8 -status
```

The output of this command can be seen in appendix 3. The test finds a front-end controller in VME slot 8 and shows the different register values for it. There is only one control board and CCU chip in the set-up and the program finds it from address 0x50, which means address 80 in decimal. The values of different status and control registers are also shown in the output. The status register A indicates that there have been no errors as the value is 0x00. Control registers are used to manage the different CCU actions such as the alarm signals, redundancy control and channels. The meanings of the different registers and their values can be found from the CCU25 specifications. [18]

Another command that can be used for testing is

```
./ProgramTest.exe -vmecaenpci -fec 8 -scantrackerdevice
```

With RPC link system there will not be any Tracker devices found, but the CCU chips can be found if they are functional. The results of the test are in attachment 4. The CCU was found from the same address as in previous test and additionally the digital optohybrid was found from channel 0x10, which is an I<sup>2</sup>C channel. The program made also a successful PIO reset.

### 3.5 Reading test pulse data from the link board

The link boards' ability to read, store and forward the data coming to the backplane FEB connector inputs was tested with a test board that feeds LVDS level high signals to the backplane connectors. The test board was acquired from Poland. The tester is very simple as it has only an input for 3.3 VDC and LVDS drivers to drive the outputs.



Figure 28: Test pulse board connected to the link board FEB-connectors.

The test pulses were read using a program called *pulses.exe*, written by Michal Pietrusinski. The software utilizes the rate histogram (RateHisto) module in the link board FPGA. This module is also called as multichannel counter. The multichannel counters count the number of input signals simultaneously for every channel. This can be used to count the number of hits in every RPC strip. The multichannel counters can be used for observing the signals either in Full Window or in Adjustable Window.

The signals coming from the front-end boards are synchronized to the LHC clock. This synchronization is performed in Synchronization Unit on the link board FPGA. The different windows mean time periods, in which the RPC signals are accepted and sent to the trigger boards. The full window has a constant width of 25 ns, but has a possibility to adjust the phase with respect to the main clock. The adjustable time window can be

adjusted in range 0-25 ns. The clocks are acquired from the TTCrx chip mounted on the link board. The rate of chamber noise can be reduced by properly setting the position and width of the synhronization window. [19]

The results of the test pulse reading can be seen in appendix 4. The hexadecimal numbers represent the number of hits per every channel. As the test pulse board did not include the possibility to set the frequency or duty cycle of the test signals, the number of hits per channel is very high. In this case the number of hits per channel should be the same on every channel, but it is not. This can result from many reasons. The lengths of paths from the input connectors to the FPGA were different for every channel in the link board prototypes, which results in different skews between the channels. This is corrected in the final design. The early version of the test pulse program did not properly utilize the Synchronization Unit functions so that all signals might not be in exactly the same time window. The test pulses can be seen in the FEB inputs however and can be read to PC with the front-end controller.

### 3.6 The link board tester

The test set-up described in this work can be used to test the main functionalities of the control board and link board. However, these tests cannot necessarily point out the exact location of the faulty signal bus or device. There are many possible problem locations on the control board and link board such as the FPGAs, optical components, ASICs and so on. The problem can be even in the frontplane, although it does not include any active components. Arja Korpela from Lappeenranta University of Technology has designed the hardware for a link board tester for her Master's thesis.

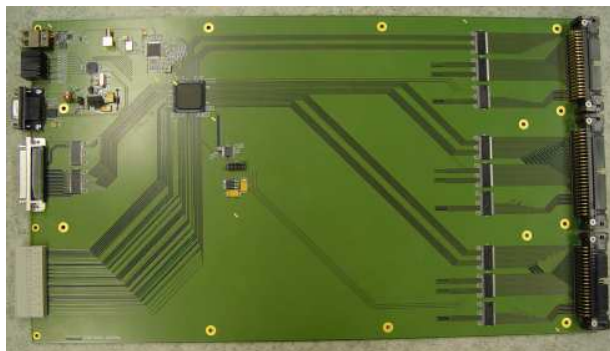


Figure 29: The link board tester.

The test board has the same connectors as the link board, so each connector can be probed with test pulses and the data can be checked. The generated test pulses can be either static bit patterns or bit shifting with logical '1' or '0'. The purpose is to make sure that the bit patterns do not change between the test board and the link board because of board defects.

The main components of the test board are Philips LPC2104 ARM microcontroller and Xilinx XC3S1000 FPGA. The microcontroller is an interface between the RS-232 and the FPGA. The testing is controlled with a user input from a terminal program and the test results will be seen in the terminal window as well as from the board front panel leds. The actual test routines are carried out by the FPGAs in the test board and the link board after the user has activated the testing from the terminal and the microcontroller has sent a logic '1' signal to the FPGA. When the tests are finished and the results can be read from the FPGA pins the microcontroller will be notified by a "test ready signal". The microcontroller reads the result pattern from five FPGA output pins and sends the corresponding text strings to the terminal. The tests cover the FEB connectors, GOL, CSC data connector and the front plane connector. The test flow is described in figure 30.

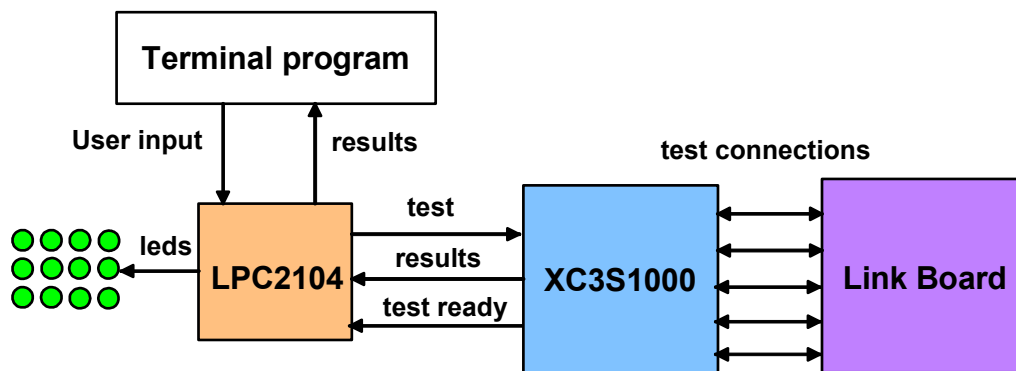
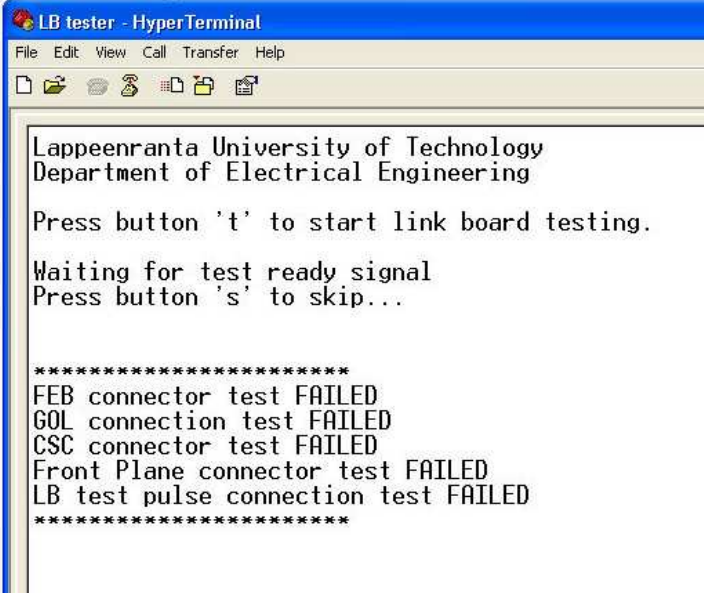


Figure 30: Test flow with the link board tester.

A sample test run without a link board attached is seen in figure 31. As there is no link board to respond to the test pulses the test board FPGA pulls the result pins in low state and all the tests are shown as failed. The test board front panel has six leds to represent each test result. If the tests are OK the five first leds is lit and the sixth led indicates that

the link board was OK. In case of a failure the corresponding leds will flash for a while and then remain off.



```

LB tester - HyperTerminal
File Edit View Call Transfer Help

Lappeenranta University of Technology
Department of Electrical Engineering

Press button 't' to start link board testing.

Waiting for test ready signal
Press button 's' to skip...

*****
FEB connector test FAILED
GOL connection test FAILED
CSC connector test FAILED
Front Plane connector test FAILED
LB test pulse connection test FAILED
*****

```

Figure 31: A sample test run without a link board attached.

The tester and the link board to be tested are placed in the same link board box with front plane attached. The test is initiated from the terminal and when the results are ready the link board is replaced with another one. The link board FPGA chips need to be programmed with separate test codes in order to get the test functionality.

## 4. Conclusions

Finding all the necessary components for the test set-up was not an easy task. There are very limited amount of prototypes or final versions available for test purposes and the designs have been in constant development. The integration of hardware and software under frequent changes can result in compatibility problems. The latest designs of the link board and control board were slightly delayed and the tests had to be done with older prototypes. The main functions are the same however, so there is no need for major changes in the test set-up.

All the devices used in the set-up were functional and the communication between software and hardware was successful. The test setup can be developed further by connecting the distribution and front-end board to the backplane. This allows us to test

the FEB reaction to link board test pulses and to test the I<sup>2</sup>C communication between the control board and FEB. As the new board designs are ready and installed in the LBx, the prototype cards used in the set-up will be replaced with them. Testing a link board box with full number of control and link boards brings new aspects to the set-up. The amount of data transfers between the boards and DCS increase significantly and one extra matter to notice is the power consumption and heat.

## References

- [1] Introducing the little Higgs. Web document.  
<http://physicsweb.org/articles/world/15/11/3> (referred 16.11.2005)
- [2] CMS Outreach – Transparencies and publicity. Web document.  
<http://cmsinfo.cern.ch/Welcome.html/CMSdocuments/CMSdocuments.html>  
(referred 10.11.2005)
- [3] Ungaro, Donatella. The Link-Board Control in the RPC Trigger System for the CMS Experiment. Helsinki Institute of Physics 2004. 148 s. ISBN 952-10-1685-X.
- [4] Fonte P. Applications and New Developments in Resistive Plate Chambers. IEEE transactions on nuclear science, Vol. 49, No. 3, June 2002.
- [5] Ammosov V.V. et.al. Study of Avalanche Mode Operation of Resistive Plate Chambers with Different Gas Gap Structures. IHEP Preprint 98-51. Protvino 1998.
- [6] CMS Collaboration. CMS TDR: the Trigger System. CERN-LHCC-2000-038.
- [7] Loddo, Flavio. RPC electronics status. Web document.  
[http://hep.fuw.edu.pl/cms/esr/talks/warsaw\\_2003\\_loddo.ppt](http://hep.fuw.edu.pl/cms/esr/talks/warsaw_2003_loddo.ppt) (referred 18.11.2005)
- [8] RPC Link Box Control System for RPC detector in LHC experiment. Web document. <http://www.ise.pw.edu.pl/~wzab/artykuly/rlbcs.pdf> (referred 16.11.2005)
- [9] Taylor B.G. Timing Distribution at the LHC.  
<http://www.cern.ch/TTC/LECC02.pdf> (referred 18.11.2005)
- [10] CERN news bulletin archive. Issue no. 51/2004. Web document.  
<http://bulletin.cern.ch/eng/articles.php?bullno=51/2004&base=art> (referred 18.11.2005)
- [11] B.G. Taylor. TTC machine interface (TTCmi) User Manual. Web document.  
<http://ttc.web.cern.ch/TTC/TTCmiManual.pdf> (referred 14.11.2005)



- [12] Overall TTC System Architecture. Web document.  
<http://www.cern.ch/TTC/WWWstuff/Block4.pdf> (referred 14.11.2005)
- [13] Frequently Asked Questions about VMEbus. Web document.  
<http://www.vita.com/vmefaq.html> (referred 10.11.2005)
- [14] TTCvi technical manual. Web document.  
<http://www.cern.ch/TTC/TTCviSpec.pdf> (referred 10.11.2005)
- [15] TTCrx Reference Manual. Web document.  
[http://ttc.web.cern.ch/TTC/TTCrx\\_manual3.8.pdf](http://ttc.web.cern.ch/TTC/TTCrx_manual3.8.pdf) (referred 10.11.2005)
- [16] Drouhin F, Gross L, Vintache D, Marchioro A, Paillard C, Ljuslin C, Mirabito L, Verdini P.G. The CERN CMS Tracker Control System. Nuclear Science Symposium Conference Record, 2004 IEEE Volume 2, 16-22 Oct 2004.  
Page(s):1196 - 1200 Vol. 2. Digital Object Identifier  
10.1109/NSSMIC.2004.1462417.
- [17] Schwick, C. Hardware Access Library Users Guide. Web document.  
<http://cmsdoc.cern.ch/~cschwick/software/documentation/HAL/manual/HALUsersGuide.pdf> (referred 11.11.2005)
- [18] Marchioro A, Ljuslin C, Paillard C. CCU25 Communication and Control Unit ASIC for Embedded Slow Control. Web document. <http://proj-fec-ccs.web.cern.ch/proj%2DFEC%2DCCS/Documents/CCU25Specsv2-1.pdf>.  
(referred 15.11.2005)
- [19] Bunkowski K, Colaleo A, Giuseppe I, Iskanian M, Krolikowski J, Kierzkowski K, Kudla M, Loddo F, Pietrusinski M, Pozniak K, Ranieri A, Tuuva T, Ungaro D, Wrochna G, Zabolotny W. Diagnostic System of the RPC Muon Trigger of the CMS Detector – Design and Test Beam Results. Web document.  
<http://ieeexplore.ieee.org/iel5/9892/31433/01462073.pdf?arnumber=1462073>  
(referred 16.11.2005)

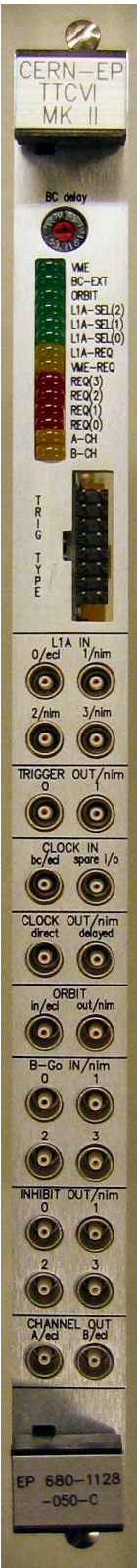


Figure 1: Front panel of the TTCvi.

## APPENDIX 2



Figure 1: Front panel of the TTCvx.

```
./ProgramTest.exe -vmecaenpci -fec 8 -status
```

```
VME FEC will be used with the file
/usr/local/xdaq/TrackerOnline/2005/FecSof
eV3_0//config/FecAddressTable.dat
Make the configuration for
- Display status for device driver / FEC / CCU
Press <Enter> to continue ...
```

```
-----
Scan the device driver FECs
```

```
Found a FEC device driver on slot 8.0
    FEC status register 0: 0xffff
        FIFO transmit running
        FIFO receive running
        FIFO receive halt full
        FIFO receive full
        FIFO receive empty
        FIFO return half full
        FIFO return full
        FIFO return empty
        FIFO transmit half full
        FIFO transmit full
        FIFO transmit empty
        Link initialise
        Pending irg
        Data to FEC
        TTCRx ok
    FEC status register 1: 0x0
    FEC control register 0: 0x0
        FEC disable
        Output ring A
        Input ring A
        Sel irq mode = 0
    FEC control register 1: ffff
```

```
Press <Enter> to continue
```

```
-----
Scan the ring to find CCUs for FEC 8 ring 0
```

```
FEC 0x8 Ring 0x0 CCU 0x50 found
    CCU status register A: 0x0
        CCU status register A = 0x00
    CCU status register B: 0x3a
        Last correctly received transaction number (SRB) 58
(0003A)
    CCU status register C: 0x0
        CCU status register C = 0x00
        Input A Output A
    CCU status register D: 0x0
        Source field for the last ring message addressed to
this CCU
0000
    CCU status register E: 0x0
        CCU status register E = 0x000000
        List of busy channels:
        - i2c:
    CCU status register F: 0x0
        Number of parity error since the last reset: 0
    CCU control register A: 0x0
        CCU control register A = 0x00
```

```
CCU control register B: 0x30
    CCU control register B = 0x30
    (RTRY) Retry count: 0
CCU control register C: 0x0
    CCU control register C = 0x00
    Input A Ouput A
CCU control register D: 0x0
    CCU control register D = 0x00
    Broadcast class: 0x0000
CCU control register E: 0x0
    CCU control register E = 0x000000
    List of enable channels:
    - i2c: None
    - PIA: None
    - Memory channel not enabled
    - Trigger channel not enabled
    - JTAG channel not enabled
```

```
./pulses.exe
```

```
FEC 8 ring 0  
CR0 = 0x0  
SR0 = 0x3c80  
SR1 = 0x8
```

```
crate...ok  
board...ok  
Starting diagnostics
```

```
Diagnostics stopped
```

```
0: a05925  
1: a05486  
2: a05993  
3: a056a0  
4: a05f32  
5: a059e3  
6: a05c2a  
7: a05e79  
8: a05e08  
9: a06242  
10: a05da1  
11: a064af  
12: a05fe2  
13: a062d8  
14: a05dea  
15: a06585  
16: a05bc3  
17: a0607f  
18: a05edf  
19: a06046  
20: a05c7b  
21: a05d3d  
22: a05b88  
23: a05b47  
24: a05dad  
25: a05973  
26: a0573c  
27: a05ae7  
28: a05675  
29: a05616  
30: a055bd  
31: a05618  
32: a05641  
33: a0529e  
34: a05325  
35: a04fd2  
36: a0523f  
37: a04f9a  
38: a04e82  
39: a04b7c  
40: a04aaa  
41: a04aaf  
42: a04abd  
43: a04b07
```

44: a04bd9  
45: a04aea  
46: a04a03  
47: a04aa4  
48: a04dee  
49: a04c79  
50: a04d95  
51: a04eec  
52: a0520a  
53: a05319  
54: a054b6  
55: a0541c  
56: a05aa1  
57: a05910  
58: a05b79  
59: a05bc9  
60: a0601b  
61: a0603f  
62: a06311  
65: a06844  
66: a06644  
67: a0684d  
68: a065b6  
69: a06891  
70: a067f4  
71: a06b0b  
72: a06447  
73: a06877  
74: a0644f  
75: a06b6a  
76: a06355  
77: a068f5  
78: a06410  
79: a06607  
80: a0635d  
81: a0635f  
82: a064a7  
83: a06432  
84: a0627a  
85: a05f99  
86: a06380  
87: a05dc8  
88: a061be  
89: a05bb0  
90: a06170  
91: a058a0  
92: a05b4b  
93: a055e2  
94: a059f3  
95: a0561b  
Done